实验12

1. **创建并编写 message\_queue.c文件**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/msg.h>

#include <time.h>

#define MSG\_SIZE 128

typedef struct {

    long mtype;

    int data;

    char text[MSG\_SIZE];

} Message;

int msgid;

void process\_A() {

    srand(time(NULL) + getpid());

    int count = 0;

    while(count < 3) {

        // 生成随机数

        int a = rand() % 3 + 3; // [3,5]

        // 发送给B

        Message msg;

        msg.mtype = 1; // B的类型

        msg.data = a;

        sprintf(msg.text, "A->B: 计算1-%d的和", a);

        msgsnd(msgid, &msg, sizeof(Message)-sizeof(long), 0);

        printf("%s\n", msg.text);

        // 等待B的回复

        msgrcv(msgid, &msg, sizeof(Message)-sizeof(long), 2, 0);

        printf("B回复A: 1-%d的和是%d\n", a, msg.data);

        // 发送给C

        msg.mtype = 3; // C的类型

        msg.data = a;

        sprintf(msg.text, "A->C: 计算1-%d的积", a);

        msgsnd(msgid, &msg, sizeof(Message)-sizeof(long), 0);

        printf("%s\n", msg.text);

        // 等待C的回复

        msgrcv(msgid, &msg, sizeof(Message)-sizeof(long), 4, 0);

        printf("C回复A: 1-%d的积是%d\n", a, msg.data);

        printf("===== 第%d次任务完成 =====\n", count+1);

        sleep(1); // 避免太快

        count++;

    }

    // 发送结束信号

    Message end\_msg;

    end\_msg.mtype = 1; // 发给B

    end\_msg.data = -1;

    sprintf(end\_msg.text, "结束");

    msgsnd(msgid, &end\_msg, sizeof(Message)-sizeof(long), 0);

    end\_msg.mtype = 3; // 发给C

    msgsnd(msgid, &end\_msg, sizeof(Message)-sizeof(long), 0);

}

void process\_B() {

    Message msg;

    while(1) {

        msgrcv(msgid, &msg, sizeof(Message)-sizeof(long), 1, 0);

        if (msg.data == -1) break; // 结束信号

        printf("%s\n", msg.text);

        int sum = 0;

        for(int i=1; i<=msg.data; i++) {

            sum += i;

        }

        msg.mtype = 2; // 回复给A

        msg.data = sum;

        sprintf(msg.text, "B->A: 1-%d的和是%d", msg.data, sum);

        msgsnd(msgid, &msg, sizeof(Message)-sizeof(long), 0);

        sleep(1);

    }

}

void process\_C() {

    Message msg;

    while(1) {

        msgrcv(msgid, &msg, sizeof(Message)-sizeof(long), 3, 0);

        if (msg.data == -1) break; // 结束信号

        printf("%s\n", msg.text);

        int product = 1;

        for(int i=1; i<=msg.data; i++) {

            product \*= i;

        }

        msg.mtype = 4; // 回复给A

        msg.data = product;

        sprintf(msg.text, "C->A: 1-%d的积是%d", msg.data, product);

        msgsnd(msgid, &msg, sizeof(Message)-sizeof(long), 0);

        sleep(1);

    }

}

int main(int argc, char \*argv[]) {

    if (argc != 2) {

        printf("Usage: %s <A|B|C>\n", argv[0]);

        exit(1);

    }

    // 创建消息队列

    key\_t key = ftok("message\_queue.c", 'a');

    msgid = msgget(key, IPC\_CREAT | 0666);

    if (msgid == -1) {

        perror("msgget");

        exit(1);

    }

    if (strcmp(argv[1], "A") == 0) {

        process\_A();

    } else if (strcmp(argv[1], "B") == 0) {

        process\_B();

    } else if (strcmp(argv[1], "C") == 0) {

        process\_C();

    } else {

        printf("Invalid argument. Use A, B or C.\n");

        exit(1);

    }

    // 删除消息队列

    msgctl(msgid, IPC\_RMID, NULL);

    return 0;

}

文本

AI 生成的内容可能不正确。

1. **编译**

图片包含 文本

AI 生成的内容可能不正确。

1. **运行三个终端窗口，分别启动A、B、C进程**

图形用户界面, 文本

AI 生成的内容可能不正确。

图形用户界面, 文本

AI 生成的内容可能不正确。